Comparing Accessibility Evaluation Plug-ins

Tânia Frazão

tfrazao@lasige.di.fc.ul.pt

LASIGE, Faculdade de Ciências, Universidade de Lisboa
Lisboa, Portugal

Carlos Duarte

caduarte@campus.ul.pt

LASIGE, Faculdade de Ciências, Universidade de Lisboa
Lisboa, Portugal

Figure 1: Screenshot of all tools studied. First row, from left to right: Microsoft's Accessibility Insight, ACCESS Assistant Community, ARC Toolkit, aXe Chrome Plug-in. Second row, from left to right: Lighthouse, Tenon, TotalValidator, WAVE.

ABSTRACT

This article reports the results of a study comparing evaluation accessibility plug-ins extensions for the Chrome web browser. Eight of the most well-known tools among developers were chosen. All tools are free or available under an open-source license, and work with the Chrome browser. The tools were compared based on their feature set, their usability and their evaluation results of ten of the Alexa top websites. We found that individual tools still provide limited coverage of the success criteria; the coverage of success criteria varies quite a lot from evaluation engine to evaluation engine; what are the most and least covered success criteria in automated evaluations. After analysing the results, we highly recommend to use more than one tool (with a different engine) and to complement automated evaluation with manual checking.

CCS CONCEPTS

• **Human-centered computing** → **Accessibility design and evaluation methods**; **Accessibility systems and tools**.

KEYWORDS

accessibility, evaluation, automatic, tools

ACM Reference Format:

Tânia Frazão and Carlos Duarte. 2020. Comparing Accessibility Evaluation Plug-ins. In *17th Web for All Conference (W4A '20)*, April 20–21, 2020, Taipei, Taiwan.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

W4A '20, April 20–21, 2020, Taipei, Taiwan

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-7056-1/20/04.

<https://doi.org/10.1145/3371300.3383346>

Taiwan. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3371300.3383346>

1 INTRODUCTION

To know if a website is accessible it needs to be evaluated. The principal objective in accessibility testing is to find if all people can use a website and give useful feedback to promote future design and implementation changes that improve the site's usability. "Test early, test often" is an old software engineering saying. To ensure quality on time and budget, accessibility evaluations should start right at the beginning of product design and be continued in following development iterations through to the final delivery [16].

Various accessibility evaluation methods are based on the use of guidelines. Over the years, the W3C Web Accessibility Initiative (WAI) has generated sets of guidelines to help systematize what is required to produce and render accessible web content [20]. These guidelines on accessibility are available in the form of checklists with hyperlinks to explanations and testing methods. The current set of guidelines for assessing the accessibility of websites is WCAG 2.1 [15] which is organized around four accessibility principles:

- **Perceivable**, which means that everyone should be able to perceive the information. Websites should not have any information that is hidden to a particular class of users;
- **Operable**, which means people need to be able to access all information and all website regardless of disabilities;
- **Understandable**, which means all the information needs to be understood by the user and the website should be predictable;
- **Robust**, which means the website should behave in a similar way regardless of the user agent that is being used to render it and still be accessible to everyone.

Each principle has a set of testable success criteria. The WCAG 2.1 extended the number of success criteria on WCAG 2.0 by 17,

taking the total to 78 testable success criteria. In order to meet different levels of requirements, each success criteria is assigned to a conformance level. There are three conformance levels: A (lowest), AA, and AAA (highest). Each level represents an increasing level of accessibility. Conformance at higher levels indicates conformance at lower levels also. For example, a web page that conforms to AA, meets both the A and AA conformance levels. There are few studies investigating the impact of the level of conformance of websites. However, it seems implementing recommendations for a high level of web accessibility leads to improved accessibility [6, 11]. Furthermore, a high level of conformance proves to benefit both users with disabilities and non-disabled users [13].

Web accessibility evaluation is an assessment procedure to analyze how well the web page can be used by people with different disabilities. Optimal results for accessibility evaluation are achieved using different approaches taking advantage of specific benefits from each of them. The approaches range from automated tool checking, to manual inspection from accessibility experts, or empiric evaluations. Automated tools can be used in isolation, but manual inspections are often supported by some kind of automated mechanisms. The focus of this paper will be on the quality of automated accessibility evaluations provided by different tools. We begin by presenting related work on accessibility evaluation, covering the problems of relying on automated evaluations and the different options to evaluate the accessibility of a web site. We then describe the experimental study comparing eight automated accessibility tools. We discuss our findings related to the tools' usability and their evaluation coverage, before concluding the paper.

2 RELATED WORK

2.1 Problems with Guidelines

Different authors argue there are problems with the web accessibility guidelines [2, 7, 9, 10, 14]. One of the issues is the relation between guideline violations and real accessibility problems. Some studies found a that evaluating the accessibility of web pages just by checking conformance with the WCAG results in overlooking several accessibility issues. This problem is aggravated if the accessibility check relies solely on automated tests which are unable to validate the full number of success criteria in the guidelines.

Another perspective that has been critiqued by practitioners is the applicability of the guidelines. Although guidelines would seem to present objective criteria against which to evaluate a system, they raise a number of difficulties. A large number of guidelines need much effort to learn and apply properly. For an accurate evaluation, every page should be evaluated against every applicable guideline, which would be very time-consuming. Following guidelines helps to improve a website, but guidelines are generalizations so there may be particular circumstances where guidelines conflict or do not apply and may be interpreted subjectively. Expertise in the evaluation of detailed characteristics is needed to apply accessibility guidelines appropriately. However, this evaluation alone is not sufficient, as it does not provide information to accurately predict user behavior. To be sure about the outcomes, evaluation of web accessibility also requires manual inspection and testing with users.

2.2 Accessibility Evaluation Methods

2.2.1 Automated Checking. Although automated accessibility checking has its role in the evaluation of websites, its strengths and weaknesses need to be understood because many WCAG checkpoints cannot be checked automatically [7]. According to Vigo et al. [14], evaluation tools are employed to reduce the burden of identifying accessibility barriers. However, an over-reliance leads to placing aside additional testing that necessitates expert evaluation and user tests. In a study about the sole reliance on automated tests, the researchers investigated the effectiveness of 6 state-of-the-art tools (AChecker, SortSite, Total Validator, TAW, Deque, and AMP). These tools had in common their capability to test web pages against the WCAG 2.0 guidelines. The coverage obtained was very narrow as, at most, 50% of the success criteria were covered. The most frequently violated success criteria (SC) were “1.3.1 Info and Relationships”, “1.4.3 Contrast”, “1.1.1 Non-text Content”, “1.4.4 Resize Text” and “2.4.4 Link Purpose”. Therefore, relying on just automated tests entails that 1 of 2 success criteria will not even be analyzed [14].

Using only automated tools is not by itself a viable answer to the problem of evaluating accessibility. The W3C/WAI states: “Web accessibility evaluation tools can not determine the accessibility of Web sites, they can only assist in doing so.” [19]

Even if automated checking tools are not sufficient to determine the accessibility or a resource, these tools are great to help developers meet accessibility standards. Developing better automated or semi-automated tools is important. With the ever-growing dynamics of web pages (e.g., through AJAX and other JavaScript techniques), the state of a web page's content, structure, and interaction capabilities might become different when compared to the initial HTTP communication. Several dynamic content techniques allow for displaying or hiding information, injecting new content, and even removing content from Web pages. It is imperative for automated evaluation to be applied to the content web browsers display [4] which can be an advantage for tools that are integrated directly on the browser. A list of automated accessibility checking tools is maintained on the WAI website¹.

2.2.2 Expert evaluations. Expert testing is important because experts understand how the underlying web technologies operate and can act as intermediaries for knowledge about different user groups [2]. Expert evaluations should be done when initial prototypes are available. They serve to identify any accessibility issues in order to eliminate them before carrying user-based evaluations. These evaluations are also done because it may not be possible to obtain actual users for evaluations or because there is insufficient time for testing with users. Expert-based methods require one or more accessibility domain experts to work through a website looking for accessibility problems [2, 7, 17]. Expert testing includes:

Tool-guided evaluation where a tool looks for accessibility problems and presents them to the evaluator. While beginners may be especially dependent on tool-guided evaluation, evaluators of all levels of experience know that even a single checkpoint may require several tests to check whether it has been passed, some of which can be automated and some of which cannot. An automated tool can check whether there is an alternative description of every

¹<https://www.w3.org/WAI/ER/tools/>

image, which can be a very useful function in evaluating. However, no automatic tool can check whether the alternative descriptions are accurate and useful [8].

Screening and using end-user assistive technology: Experts can work through task scenarios that represent what users would typically do [7]. Screening means experts try to reproduce the experiences of people with disabilities. They can use assistive technology to interact with a site or try to restrict one's abilities in some manner [17].

Structural inspection where the evaluator uses a tool to probe how the various components of a web site work together. Inspection tools are designed to review the structure of web content. By definition, structures define what the components of a web page are and how they are related to one another. Experts can, by using structural inspectors, check the HTML DOM (document object model) which is parsed by the browser. The browser associates different behaviors with particular components. Common assistive technology does not process web document object models directly, they utilize the browser's and plugin's representation of web content in terms of structural systems. By using the DOM inspectors these show the tree of elements, attributes, and text composed out of the (X)HTML serialization, whereas the web accessibility inspectors abstract distinct components or relationships and list them [16].

Code review where the evaluator looks directly at the code and assets of a web site to search for problems. After certain problems are resolved by using a checker tool, the experts can move on to manual testing and do a detailed inspection.

From the above it becomes apparent that even for expert evaluation, automated checking tools are paramount. For that reason, we studied a subset of automated tools in order to understand the levels of support currently offered by such tools.

3 EXPERIMENTAL STUDY

This paper presents an experimental study of eight automated web accessibility evaluation extensions for the Google Chrome browser: aXe Chrome Plugin, Tenon Check, Wave Chrome Extension, TotalValidator, ACCESS Assistant Community, Microsoft Accessibility Insights and ARC Toolkit (figure 1). These represent the most used tools among developers that can be freely accessed through the Google Web Store. The initial tool selection criteria included, besides being freely accessible, that they were available as browser extensions. As depicted earlier a web page is represented not just by its HTML source, but also by a set of ancillary resources that are processed by the browser and transform the contents of the DOM. By using a browser plug-in to drive the accessibility evaluation, the evaluator can expect that the content that is evaluated is not only what is presented to the user, but also that it is evaluated in the same conditions in which it is consumed by the user (e.g., viewport size). Also, these tools might fit better into an expert evaluation workflow by providing support to the evaluator directly in the browser instead of requiring the use of additional applications.

This study centers on evaluating differences in the tools, how they are implemented, what success criteria they use, how many tests for each success criteria, the overall differences in the results of the reports generated, if they perform only automated tests or if they guide the developers through manual testing. This is important

Table 1: Classification (0 min, 5 max) and number of users of automated tools according to Chrome Web Store (data collected on Dec, 4, 2019).

Tool	Classification	Users
Lighthouse - accessibility	4.5	455,902
WAVE Chrome Extension	4.2	186,810
aXe Chrome Plugin	4.6	102,196
Accessibility insights	4.6	34,613
Arc toolkit	4.0	5,897
TotalValidator	2.3	5,239
ACCESS Assistant Community	4.6	3,059
Tenon Check	4.8	2,191
Average	4.2	99,488.38

to understand the impact of automatic evaluations of the accessibility of web pages in the browser environment. Consequently we analyze the tools differences, strengths and limitations according to their evaluation results but also taking into account their usability. With the information gathered, we hope to contribute to future improvements in this class of tools.

3.1 Tools

The criteria used to select the automated tools considered the type of tool, in this case, it needs to: be available as an extension for the Google Chrome Browser, although some are also available for the Mozilla Firefox Browser; be available for download in the Chrome Web Store and have a free or open-source license. Since this resulted in a large list of tools, we considered only the tools that are most prevalent among users. We chose the final list according to their popularity, classification, and number of users. As can be seen in table 1, the most used tool is Lighthouse. Interestingly, this is the only of the studied tools that is not listed in the W3C list of web accessibility evaluation tools [18]. It is important to mention that most of these tools have upgrade products available except Microsoft Accessibility Insights, Lighthouse, and Wave. In the following paragraphs we briefly introduce the selected tools.

3.1.1 Accessibility Insights. Microsoft Accessibility Insights supports two primary workflows: FastPass and Assessment. The FastPass workflow helps developers identify common high impact accessibility issues. FastPass begins with automated checks. Failures are highlighted directly in the target page. Clicking a failure shows details including how to fix it and the corresponding errors in the DOM. FastPass has also an assisted manual test for tab stops explaining what problems to look for with the aid of a visual helper. The second workflow is Assessment. It helps anyone with HTML skills to conduct a thorough accessibility evaluation. It has approximately 20 manual tests with test instructions. It has also detailed information about dos and don'ts, how to fix and links to WCAG success criteria, techniques, and common failures. The automated tests with the manual assessment provide WCAG 2.0 AA coverage.

3.1.2 ACCESS Assistant Community. With over 100 fully automated accessibility checks, Assistant Community runs a Quick Test on any web page to view details for all accessibility violations on the page, along with remediation guidance. This extension accesses

URLs for all open tabs to enable the testing of pages on any open tab. Preview Modes features a guide for manual testing efforts. Each preview mode applies markup or styling to the page to help testers identify common accessibility violations at a glance.

3.1.3 ARC Toolkit. The ARC Toolkit uses the automated ARC Rules. It has a sidebar which shows ARC rules engine assertions organized in categories and sub-categories. Types of results are split into visible and hidden errors and warnings. Hidden are not visible in the browser but may impact assistive technology users such as options in a menu that only appear when the menu is open. Warnings are potential issues that have been flagged but require manual verification. The test results show code followed by rules and assertions for the issue, along with a brief description and recommendation. Additional features complement a manual accessibility check process. For example, the “show and track focus” checkbox can enhance the visual focus with a bold red outline. It also has a feature for tab order visualization. When selected the toolkit shows a representation of what a keyboard user experiences when relying on the TAB key to navigate to each active element. It has a possibility to submit the DOM or URL to the W3C Markup Validation Service.

3.1.4 aXe Chrome. aXe Chrome shows the user the accessibility violations and their number of occurrences on the page. Each violation includes the actual description of what is wrong and, if clicked, opens a page from Deque University that contains an even more detailed description. Under the actual issue description there is also an excerpt of the code that is causing the issue, information on how it can be fixed and also an indication if it is a critical issue. The highlight option will highlight the offending component on the page.

3.1.5 Lighthouse. Integrated in Dev Tools of the Chrome browser, Lighthouse is accessed through the “Audits” option. Lighthouse gives the error results and, after clicking, explains more about the error and how to fix it. It provides additional items to manually check. These items address areas which an automated testing tool cannot cover. It also informs the user about instances that have passed and gives also information about not applicable tests.

3.1.6 Tenon Check. Tenon Check operates based on an API and requires the user to sign up and obtain an API key and then configure the extension with the API key obtained. After clicking the extension button, Tenon asks the user to sign-in and after that the web page is evaluated. The Tenon website then shows a “prettified” version of the JSON received after the request triggered by clicking the extension button, together with a dashboard with the results. It presents how many issues were found, the URL, and other information. The left graph on the dashboard shows the number of tests that were run and how many passed. The right graph shows the relative percentage of failures. The results show code snippets which is actual HTML from the DOM. The Tenon specifications consider an issue an error when it reaches 80 percent of certainty. The priority score presented is normalized considering other issues on the page. Users are able to download the results in csv format.

3.1.7 TotalValidator. After loading the page, the user clicks on the extension and it opens the pre-installed Total Validator on the

Table 2: Web sites selected for the evaluation (data collected on Dec, 4, 2019).

#	Site	Language
1	google.com	English
2	youtube.com	English
3	tmall.com	Chinese
4	wikipedia.org	English
5	yahoo.com	English
6	amazon.com	English
7	reddit.com	English
8	live.com	English
9	netflix.com	English
10	blogspot.com	English

system. The user has the option to first choose against which guidelines the page should be evaluated. The browser version presents a summary of the issues it finds (total errors and total warnings). In the free version only the id of the issue is shown (a number that identifies the issue in the TotalValidator documentation). Clicking on the id number opens the documentation explaining in more detail what the violation is and the success criteria it violates.

3.1.8 WAVE. This tool presents the results showing the page with embedded icons and indicators using a color system. Red icons indicate accessibility errors that need to be fixed. Green icons indicate accessibility features – things that probably improve accessibility (though these should be verified). The other icons and indicators, particularly the yellow alert icons, highlight other elements that the user should evaluate. The sidebar has a summary of the errors that were detected, alerts and structure elements. It also provides a manual contrast testing: by changing the color, the tool immediately tells us if it passes contrast checks. There is a “No Styles” option that disables all styles in the original page. This can be helpful in pinpointing where errors occur and in ensuring that the reading order of the page is logical. Users can view a brief overview of what each icon means by clicking it and viewing its documentation.

3.2 Selection of Web Sites

To compare the coverage of the different tools, we picked a sample of web sites from the Alexa Top Sites list². From each web site we used its home page. We discarded web sites where the home page was simply a log in page (e.g. Facebook). From the global list we selected the first 9 web sites in English and the first non-English language web site that met the aforementioned criteria. The sample of web sites is presented in table 2.

3.3 Testing Environment

The testing environment uses a Chrome extension to trigger the execution of the evaluation within the browser. The version of the browser used is Version 78.0.3904.108 (Official Build) (64-bit). In order to have a fair comparison, two requirements were raised:

- (1) All tools should evaluate the same web page. Given that all the evaluated sites are dynamic, we started the evaluation for

²<https://www.alexa.com/topsites>

all the tools within an interval of few seconds, and inspected the loaded pages to ensure there were no differences between them.

- (2) There should be no external interference changing the page's DOM, apart from eventual scripts of the testing tool. To meet this requirement, we created eight sandboxes. In each sandbox the only installed software was the browser (in the default configuration) with the corresponding extension (plus any additional software required by the extension). For the sandboxes we used Sandboxie (version 5.31.6).

3.4 Procedure

Before starting the evaluations we gathered information about the success criteria each tool verifies. Then, we loaded the first web page in all browsers. After execution of the evaluations from all the tools we collected their outputs. The same procedure was repeated for the remaining web pages. Once all results were collected we proceeded to analyze them according to the accessibility violations found and success criteria coverage.

4 RESULTS

4.1 Feature Analysis

Tables 3 and 4 present a comparison of the features of the tools evaluated. The information was gathered from the W3C list of accessibility evaluation tools, the tool's online website and our usage to evaluate the sample of web pages. Notice that there are some tools that are directly integrated into development tools of the browser and have to be accessed by clicking Inspect on the dev tools console, while others are activated when the user clicks the extension button. Only TotalValidator required the pre-installation of software outside the browser environment. When the user starts the TotalValidator extension it opens the pre-installed application. There are some tools that use the same engine, like Accessibility Insights and Axe-chrome, which both use the aXe engine.

Tools differ in specifications concerning which success criteria they evaluate, how they implement the tests to analyze each success criteria and how they classify the violation into error. On this topic, it is important to notice that one tool, Tenon, distinguishes between errors and warnings based on a predefined threshold percentage of uncertainty about the result of the evaluation. When the evaluation falls below the threshold, the tool is not sure if it identified a real issue and advises the user to do an additional manual evaluation. Some tools also classify the severity of the violation (for example, from minor to critical). This classification is based on a weight given to each test. The heavier-weighted tests have a larger impact on the overall accessibility of the web page. The weights are based on heuristics defined by the tool vendors. When tests identify a problem, some tools assist developers to correct it by giving extra information on how to solve it.

A few tools also give information about instances that are considered as "Pass", meaning that the website passed the test because the instances are present and correct. For some tools, like Accessibility Insights, they state that "Pass" also means there are "No matching/failing" instances, so the absence of a feature is considered a "Pass" in its results.

Figure 2: Semi-automated test on aXe Chrome.

Besides automatic tests, tools may provide a semi-automatic evaluation where the tests identify elements or conditions to which a success criteria applies but cannot evaluate if it is a "pass" or a "fail". In these instances, tools indicate the instance is present and ask the user to check whether there is a violation of the success criteria. Figure 2 shows an excerpt from a result report of aXe Chrome, where the tool asks the user to inspect and verify if an issue is present or not.

Another category of tests are manual checks. These are not tests made by the tools, but instead, are instructions to assist their users in conducting the evaluation, for example, to verify the site with keyboard-only navigation. The tools that provide manual procedures are Accessibility Insights, ARC Toolkit, Access Community, Lighthouse and Wave. The procedures range from manual contrast evaluation in Wave to more elaborated and guided procedures in Accessibility Insights.

Some tools also provide "Best Practice" tests. They are not checks of success criteria, but are tests to identify valid and "well-formed" HTML and CSS code. When the code is not up to standard the tools give suggestions on how to modify it. Standards abiding code is halfway to implement the WCAG standards and design accessible websites [5].

All tools have unlimited use, except Tenon that limits the number of monthly requests to the API, besides being the only tool requiring the user to sign up and obtain an API key. All the tools, except TotalValidator, do not provide information on how well they support accessibility. TotalValidator informs that the creators regularly test if the tool can be used entirely by keyboard and the results generated are accessible. It also states to support screen readers.

Table 5 presents the number of tests each tool implements grouped according to the WCAG level of conformance. This information was gathered from the documentation available for each tool online. We can see that ACCESS Assistant Community, ARC toolkit and WAVE do not evaluate AAA success criteria. For A and AA levels, ACCESS Assistant Community provides the highest number of checks.

It is also important to examine exactly what the checks validate. Figure 3 presents a chart displaying the number of success criteria each tool tests. It is interesting to note if a user employs all tools, the total of individual success criteria tested is 62, still well below the total of 78 success criteria in WCAG. This is an example of the problems of over reliance on automated tools. We also looked at which success criteria were most checked overall. Figure 4 presents a treemap where the areas are related to the number of checks, across all tools, that test that success criterion. To improve legibility only success criteria with over 15 checks were included in the treemap. The most tested success criteria belong to the Perceivable principle. The other most tested principles are Operable, Robust and

Table 3: Features of evaluated tools.

Tool	Vendor	Version (Last Update)	Compliance	Browsers	Supported Formats	Type of support	Can check	Engine
aXe Chrome Plugin	Deque Systems	4.1.0 (24th October 2019)	WCAG 2.0, Section 508	Chrome	HTML, XHTML	Generating reports.	Single web pages. Restricted or password protected pages.	axe-core
Tenon Check	Tenon LLC	2.0.4(12th October 2019)	WCAG 2.0, Section 508	Chrome	HTML, CSS	Generating reports. Ability to export report in csv format.	Single web pages. Code snippets.	tenon
WAVE Chrome Extension	WebAIM.org	3.0.3 (3rd of November 2019)	WCAG 2.1, WCAG 2.0, Section 508	Chrome, Firefox	CSS, HTML, XHTML, Images	Generating reports. Displaying information within web pages. Modifying the presentation of web pages.	Single web pages. Restricted or password protected pages. Locally stored. Highly dynamic pages.	wave
TotalValidator	Total Validator	5.0 (5th of April 2019)	WCAG 2.0, WCAG 1.0, Section 508	Chrome, Firefox	HTML, XHTML	Generating reports.	Single web pages. Groups of web pages. Restricted or password protected pages.	TotalValidator
ACCESS Assistant Community	Level Access	7.8.0.344 (31th October 2019)	WCAG 2.1, WCAG 2.0, Section 508	Chrome, Firefox	WAI-ARIA, HTML	Generating reports.	Single web pages. Restricted or password protected pages.	Level Access
Accessibility Insights	Microsoft	2.10.3 (28th October 2019)	WCAG 2.1, WCAG 2.0, WCAG 1.0 Level AA	Chrome, Edge	HTML	Generating reports. Providing step-by-step evaluation guidance. Displaying information within web pages.	Single web pages.	axe-core
ARC toolkit	The Paciello Group	3.2.0.0 (4th July 2019)	WCAG 2.0, WCAG 2.1, Section 508	Chrome	WAI-ARIA, CSS, HTML, XHTML, SVG, Images, SMIL	Displaying information within web pages.	Single web pages.	ARC
Lighthouse - accessibility	Google	5.6.0 (18th October 2019)	WCAG 2.0, WCAG 2.1, Section 508	Chrome	HTML	Generating reports.	Single web pages. Restricted or password protected pages.	axe-core

Table 4: Behavior, information and tests of evaluated tools.

Tool	Behaviour	How to fix issues	Highlights issues in the code	Severity classification	Information about the violated check	Upgrade version	Semi-Automatic evaluation	Best practice evaluation	Manual evaluation
aXe Chrome Plugin	Integrated in developer tools	Yes	Yes	Yes	Yes (plus other resources)	Yes	Yes	Yes	No
Tenon Check	Displaying information in Tenon web app	Yes	Yes	Yes	Yes	Yes	No (if using only extension) Yes (if using the Tenon website)	No	No
WAVE Chrome Extension	Click on the extension	No	No	No	Yes	No	Yes	Yes	Yes
TotalValidator	Click the extension to open TotalValidator desktop app	No (redirects to W3C)	No	No	Yes	Yes	Yes	Yes	No
ACCESS Assistant Community	Click the extension to test all open tabs	Yes	Yes	No	Yes	Yes	No	No	Yes
Accessibility Insights	Integrated in developer tools	Yes	Yes	No	Yes	No	Yes	No	Yes
ARC toolkit	Integrated in developer tools	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Lighthouse - accessibility	Integrated in developer tools	Yes	Yes	No	Yes	No	No	Yes	Yes

Table 5: Number of checks of each tool by level of conformance.

Tool	A	AA	AAA	Total
aXe Chrome Plugin	35	7	2	44
Tenon Check	103	5	23	131
WAVE Chrome Extension	152	28	0	180
TotalValidator	108	11	20	139
ACCESS Assistant Community	215	37	0	252
Accessibility Insights	55	7	2	64
ARC toolkit	136	9	0	145
Lighthouse - accessibility	39	4	4	47

Understandable by this order. Three success criteria stand out by the number of checks related to them: 1.3.1 Info and Relationships, 1.1.1 Non-text Content, and 4.1.2 Name, Role, Value. Perceivable is the principle with most checks only because of 1.3.1 and 1.1.1. In addition to those two it has only one more success criterion with more than 15 checks. The coverage of the Operable principle is more uniform, with 7 success criteria with between 16 and 51 checks. Both the Understandable and the Robust principles have only two success criteria with more than 15 checks. For the Robust principle this could be expected since it has only three success criteria (and one of those was only introduced in WCAG 2.1, which is not covered by all tools). However, the Understandable principle has 17 success criteria, hinting at the difficulty of automating these success criteria. A similar argument can be made for the Perceivable principle that has 29 success criteria with only 3 having more than 15 checks and for the Operable principle with 29 success criteria also, although for this one the situation is not as pronounced.

Figure 3: Number of success criteria tested by tool. **Figure 4: Number of checks by success criteria.**

4.2 Usability Analysis

In the following paragraphs we discuss the strengths and limitations of each tool from a usability perspective. Given the diverse users of these tools (accessibility experts, developers, monitoring agencies, etc.) that might or might not have accessibility expertise, we did not assume the needs or knowledge of an accessibility expert in our analysis. The analysis is based on the insights we gathered while using the tools and not on a formal evaluation procedure.

4.2.1 Accessibility Insights. Strengths: Easy to use; Very good user interface; Explanations about errors good and easy to follow; Manual testing procedures with visual help present; Good coverage of automated tests and approximately 20 manual guided tests. **Limitations:** Sometimes the tool crashed and it was necessary to refresh the browser.

4.2.2 aXe Chrome Plugin. Strengths: Easy access to success criteria documentation; has good material about what is an issue and how to fix it; gives the possibility to save results; good, intuitive user interface; if tests are semi-automatic asks and guides the user to evaluate the result. **Limitations:** The extension stopped working a few times, specially when testing pages dense in information.

4.2.3 ARC Toolkit. Strengths: Easy access to success criteria documentation; Includes easy tests for focus order and new WCAG 2.1 success criteria Reflow and Text spacing as well as buttons to validate the code through the W3C Nu HTML validator. **Limitations:** The user interface can be difficult to understand initially.

4.2.4 Tenon. Strengths: Errors are identified by an id; Possible to export results in CSV format; Good documentation available about how to use and success criteria evaluation. **Limitations:** Necessary configuration of API password before using; when clicking on the browser extension button it opens multiple browser windows with the reports instead of only one; results not aggregated by error.

4.2.5 WAVE. Strengths: Easy access to success criteria documentation; Simple results report; Possibility to turn-off CSS of the page; Direct contrast evaluation where user can manipulate colors and check for contrast directly with tool. **Limitations:** Uses a lot of icons on the page to help users locate instances, which makes it confusing in pages with many errors; Most of the information presented are alerts that there is content that can be a possible accessibility issue which it cannot evaluate automatically and needs user evaluation which is time consuming.

4.2.6 TotalValidator. Strengths: Easy access to success criteria documentation; Possibility for user to validate HTML; very concise information about errors and success criteria; Provides also information about best practices. **Limitations:** Difficult to work with extension because needs to pre-install an application on the computer; Often, the extension when clicked cannot connect to app to start evaluating the web page; Doesn't show the failing instances in code or how to solve them in the free version.

4.2.7 ACCESS Assistant Community. Strengths: Easy user interface; Provides manual testing guidance. **Limitations:** Very difficult to access the documentation for the extension and success criteria; Does not show the count of instances failing per error.

4.2.8 Lighthouse. Strengths: Easy access to success criteria documentation; Easy to use; Generates a report that gives information on all of the tests that passed in addition to the ones that failed. **Limitations:** Does not show the count of failing instances per error.

4.3 Web Pages Evaluation Analysis

We begin by analyzing the differences between the errors identified by each tool in the web pages evaluated. To facilitate the analysis we

Figure 5: Number of errors found by Accessibility Insights.

Figure 6: Number of errors found by ACCESS Assistant Community.

grouped the failed checks by their success criteria. In the following charts we present these results. To accommodate the large range of errors found, we use logarithmic scales in the charts.

Figure 5 presents the number of errors found by Accessibility Insights in all web pages evaluated, grouped by success criterion. Most errors were found on success criteria 4.1.1, 1.4.3 and 4.1.2. In what concerns principles and conformance levels, the errors mostly address the Robust and Perceivable principles and the level A of conformance. The web pages with more errors found by Accessibility Insights were youtube.com, reddit.com and tmall.com.

Figure 6 presents the number of errors found by ACCESS Assistant Community in all web pages evaluated, grouped by success criterion. The success criterion with more errors was 4.1.1 largely due to the evaluation of youtube.com. The success criteria 1.1.1 and 3.2.2 are the next criteria with more errors found on all pages. The principle violated more frequently is Robust, with level A conformance being dominant in the criteria violated. The web page with more errors found was youtube.com.

Figure 7 presents the number of errors found by ARC Toolkit in all web pages evaluated, grouped by success criterion. The success criteria with more errors are 1.4.3, 1.1.1 and 4.1.2. Although youtube.com is present with a high number of errors, in the 10

Figure 7: Number of errors found by ARC Toolkit.

Figure 8: Number of errors found by aXe Chrome.

websites ARC found 1.4.3 as the most common violation with more incidence for blogspot.com and amazon.com. For ARC Toolkit, Perceivable is the principle more frequently violated with several level A and AA criteria violated.

Figure 8 presents the number of errors found by aXe Chrome in all web pages evaluated, grouped by success criterion. For aXe Chrome, the most frequent violation happens for success criteria 4.1.1, 4.1.2, 1.1.1. The Robust principle and the conformance level A are the ones with most success criteria violated. The websites with more errors are youtube.com, tmall.com and reddit.com. aXe Chrome and Accessibility Insights use the same engine. Even though there are small differences the results are similar.

Figure 9 presents the number of errors found by Lighthouse in all web pages evaluated, grouped by success criterion. Lighthouse also uses the same aXe-core engine. As aforementioned, when tools use the same engine, the implementation and heuristics can be slightly different, but the results tend to be similar. In this case, the criteria found to have more errors are very similar to Accessibility Insights: 4.1.1, 1.4.3 and 4.1.2. The web pages evaluated with more errors are youtube.com, reddit.com and tmall.com. Robust is the principle more violated and A the conformance level.

Figure 10 presents the number of errors found by Tenon in all web pages evaluated, grouped by success criterion. The success

Figure 9: Number of errors found by Lighthouse.

Figure 10: Number of errors found by Tenon.

criterion with most errors found by this tool was 1.1.1 followed by 2.4.9 and 2.4.4. The principles most violated were Perceivable and Operable and the conformance level with more errors is A. Tenon evaluates conformance level AAA more frequently than the other tools. For Tenon, youtube.com is the website with more errors followed by yahoo.com and live.com.

Figure 11 presents the number of errors found by TotalValidator in all web pages evaluated, grouped by success criterion. The success criterion with more errors detected by the tool was 1.4.3 followed by 2.4.6 and 1.1.1. The website with more errors is reddit.com, followed by youtube.com and amazon.com. More errors were found for conformance level AA than A. The principles violated more frequently were Perceivable and Operable.

Figure 12 presents the number of errors found by WAVE in all web pages evaluated, grouped by success criterion. The most prevalent errors are in success criteria 1.4.3, 1.1.1, and 2.4.4. The web pages evaluated by this tool with more errors are reddit.com, youtube.com and tmall.com. The principle most often violated is Perceivable and the level conformance most violated is A. It is important to recall that WAVE does not perform level AAA evaluations.

By inspecting the numbers of violations on all pages evaluated (table 6) a tendency of the current state of automated accessibility evaluation tools can be inferred. The conformance level A is the

Figure 11: Number of errors found by TotalValidator.

Figure 12: Number of errors found by WAVE.

one with most violations found, corresponding also to a higher number of checks targeting success criteria of this level. On what concerns principles, Robust emerges as the one with more violations found. However, this is a direct result of a single page/tool combination. If the violations from ACCESS Assistant Community found on youtube.com are removed, then Principle becomes the more violated principle followed by Operable. It should be highlighted the low number of violations in the Understandable principle, a consequence of the low number of checks in this principle.

Figure 13 provides an overview of what success criteria are most violated across all the pages tested, taking into account the results of all tools. This, together with the individual outcomes of each tool, paints a picture of the current coverage of automated accessibility evaluation tools.

Analysing the success criteria evaluated by tools, there were some criteria that were unique to some tools. The unique success criteria are 3.1.2 and 3.2.2 for ACCESS Assistant Community; 2.4.3 for ARC Toolkit; 3.3.1 for aXe Chrome; 2.1.3, 3.2.4 and 2.5.5 for Tenon; 2.5.3, 2.2.2 and 1.2.1 for TotalValidator; and 3.3.3 for WAVE. Lighthouse and Accessibility Insights do not have any unique success criteria as can be expected since they share the same engine (aXe Chrome probably has a different version of the engine justifying its unique success criteria). On the other end of the spectrum,

Table 6: Number of violations reported by each tool across all pages tested, grouped by conformance level and principle (P - Perceivable; O - Operable; U - Understandable; R - Robust).

	Conformance			Principle			
	A	AA	AAA	P	O	U	R
Acc. Insights	179	75		114	26	5	109
ACCESS	4114	7		261	52	53	3755
ARC	190	110		201	40	1	58
aXe	187	30		71	24	6	116
Lighthouse	131	56		73	15	6	93
Tenon	808	110	306	574	546	49	55
TotalValidator	339	508	12	455	298	23	83
Total	5948	896	318	1749	1001	143	4269

Figure 13: Number of violations found by all tools grouped by success criteria.

the success criteria that appear in every tool are 1.1.1, 4.1.2 and 2.4.4. Success criteria 1.3.1 and 2.4.4 appear in all but one tool.

The success criteria with more errors are 4.1.1, 1.4.3 and 1.1.1. But different tools find more violations in different sets of success criteria. One distinguishing factor between tools seems to be their level of support for success criteria of the Robust principle. One group of tools composed by Accessibility Insights, aXe Chrome, Lighthouse (the three sharing the same engine) and ACCESS Assistant Community edition all find several errors for criteria 4.1.1 and 4.1.2. The remaining tools, ARC Toolkit, Tenon, TotalValidator and WAVE find a much smaller number of violations for the same success criteria. Instead, three of these tools find more violations of success criteria 1.4.3.

5 DISCUSSION

There have been discussions and arguments about how to measure the accessibility of websites [3]. Ideally, any thorough accessibility evaluation should involve automatic and manual approaches, but that it not always possible due to different constraints (e.g. time, cost, expertise). Automatic approaches use automatic tools. Some of these are available for the browser. After triggering the evaluation, the tool evaluates the web page and generates a report. An ideal report flags accessibility issues, describes them, and gives guidance on how to fix them.

In this study we analysed accessibility evaluation tools freely available as browser extensions. Browser extensions are easily accessible and do not require the installation of further software. For that reason it is paramount that these extensions are easy to use, since they can be found and installed by users without previous accessibility expertise. Such tools facilitate the workflow of developers and the general public, for example, government bodies that need to verify if public websites are corresponding to the mandatory accessibility directives.

Overall, the tools were easy to install and use but not flawless, since during evaluations they crashed and it was necessary to refresh the browser. In these instances we took care to check if the reloaded page was still unchanged when compared to the pages evaluated by the other tools. However, of the tested plug-ins, we found one that demanded the user to install an application in addition to the plug-in: TotalValidator.

The results obtained between tools varied in what success criteria they evaluate (for example, WAVE does not evaluate level AAA conformance criteria), varied in classifying the results according to its impact on the user and in classifying the result to be an error or a warning. All these variances are a consequence of how specifications and heuristics are implemented for each tool.

Our study analyzed the quantity of errors found per success criteria by the different tools. Although there were some differences in success criteria evaluated between tools, some similarities in the results could be found, with 4.1.1 being the most often violated success criteria.

Our analysis showed that individual tools have poor coverage of the WCAG 2.1 success criteria. If all are used together the coverage increases, affording approximately 10% to 40% more coverage than if using only one tool. To improve the evaluation performance, developers should fill the gaps between tools using more than one. To leverage the evaluation there are tools, for example, TotalValidator or ARC Toolkit who can perform, besides accessibility evaluation related to success criteria, a validation of the HTML code evaluation. A well written code (i.e. abiding to standards) is halfway to meet accessibility requirements.

Considering the usability issues found during this study and the results of the evaluations performed by the tools, users may struggle with accessibility evaluations because of the inherent variance between implementations, on how to use the tools and how to interpret the results. Given that the best option seems to be to use more than one tool, to improve accessibility and adherence to developing accessible websites, there should be more consensus in how tools are developed, concepts defined and results presented. In what regards concepts, semi-automatic tests need human intervention. However, this can be due to tools only being able to detect instances of elements that have to be tested without being able to evaluate if the element meets the criteria (e.g. Wave); or a tool may need human intervention because it detects a failing instance but with an uncertainty factor too high to be sure about the correctness of its decision (e.g. Tenon). Another example are the pass or fail results: some tools consider a pass even when the instance is absent (e.g. Accessibility Insights) while others, like Lighthouse, consider a pass only if the instance is present and passed the test. In this situation, the other tests that were not performed because the instance

is absent are considered “not applicable”. All these differences in concepts impact how results are presented and interpreted.

In addition to differences in how concepts are interpreted by the tools, there may also be differences in how tools interpret the WCAG success criteria. One limitation of this study is that we did not look at possible false positives or false negatives in the evaluation outcomes of the tools. However, previous studies did [1, 12] and found that tools provide different outcomes when evaluating the same resource. As mentioned, this is often a consequence of WCAG not being completely objective in their presentation of the different success criteria or techniques. One effort currently underway that aims to mitigate this problem is the creation of ACT Rules³. ACT Rules are harmonised interpretations of accessibility standards, such as WCAG and WAI-ARIA. Tools that implement ACT-Rules must report the same outcome and thus promote an easier adoption of accessibility testing methodologies.

Overall, automatic tools available as browser extensions allow users to assess accessibility in an easy way. Although not perfect and with limitations, they still remain important to help users evaluating the accessibility of websites to find most violations of success criteria that can be automated. However, they should always be complemented with manual testing procedures and the results be analysed objectively with a reasoned judgment.

6 CONCLUSIONS

Accessibility is about different and alternative forms of presenting website content for everyone including people with all ranges of disabilities. Not only it is becoming mandatory by regulators, having accessible content is considered advantageous for both users with non-disabilities and disabilities and a good value proposition for businesses that can target more clients. The W3C guidelines and standards help developers develop more accessible websites. When developing or redesigning a website or web application, evaluating accessibility early and throughout the development process is important and makes it easier to address them. There are tools easy to install and to use directly within the browser. The tools evaluated in this paper provide a good starting point for testing, point to patterns of errors that humans can then look out for, and can catch issues before they go to production. The eight tools selected evaluated the same websites, but produced differences that reflected the variance between results. For example, some tools identify more instances of violations than others for the same success criteria or analyse different success criteria. While automated tools can discover many critical issues, there are even more issues that need human analysis. In the end, to fully test for accessibility issues, a combination of automated, manual testing and critical thinking is required. Web accessibility must be seen as a goal and must be constantly improving.

ACKNOWLEDGMENTS

This work was supported by the WAI-Tools Project, co-funded by the European Commission (EC) H2020 GA 780057. This work was supported by FCT through funding of the LASIGE Research Unit, ref. UIDB/00408/2020.

³<https://act-rules.github.io/rules/>

REFERENCES

- [1] Giorgio Brajnik. 2004. Comparing accessibility evaluation tools: a method for tool effectiveness. *Universal Access in the Information Society* 3, 3 (2004), 252–263. <https://doi.org/10.1007/s10209-004-0105-y>
- [2] Giorgio Brajnik. 2008. Beyond Conformance: The Role of Accessibility Evaluation Methods. In *Web Information Systems Engineering – WISE 2008 Workshops*, Sven Hartmann, Xiaofang Zhou, and Markus Kirchberg (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 63–80.
- [3] Giorgio Brajnik, Andrea Mulas, and Claudia Pitton. 2007. Effects of Sampling Methods on Web Accessibility Evaluations. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '07)*. ACM, New York, NY, USA, 59–66. <https://doi.org/10.1145/1296843.1296855>
- [4] Nádia Fernandes, Daniel Costa, Sergio Neves, Carlos Duarte, and Luís Carriço. 2012. Evaluating the Accessibility of Rich Internet Applications. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (W4A '12)*. ACM, New York, NY, USA, Article 13, 4 pages. <https://doi.org/10.1145/2207016.2207019>
- [5] Jon Gunderson. 2009. Functional Accessibility Testing Using Best Practices. In *Universal Access in Human-Computer Interaction. Addressing Diversity*, Constantine Stephanidis (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 506–514.
- [6] Eleanor T. Loiacono and Soussan Djasasbi. 2013. Corporate website accessibility: does legislation matter? *Universal Access in the Information Society* 12, 1 (01 Mar 2013), 115–124. <https://doi.org/10.1007/s10209-011-0269-1>
- [7] Helen Petrie and Nigel Bevan. 2009. The Evaluation of Accessibility, Usability, and User Experience. In *The Universal Access Handbook* (1st. ed.), Constantine Stephanidis (Ed.). CRC Press, Boca Raton, Chapter 20. <https://www.taylorfrancis.com/books/9780429135309/chapters/10.1201/9781420064995-31>
- [8] Helen Petrie, Chandra Harrison, and Sundeep Dev. 2005. Describing images on the web: a survey of current practice and prospects for the future. In *Proceedings of the 11th International Conference on Human-Computer Interaction (HCI '05)*.
- [9] Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines Are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 433–442. <https://doi.org/10.1145/2207676.2207736>
- [10] Dagfinn Rømen and Dag Svanæs. 2012. Validating WCAG versions 1.0 and 2.0 through usability testing with disabled users. *Universal Access in the Information Society* 11, 4 (01 Nov 2012), 375–385. <https://doi.org/10.1007/s10209-011-0259-3>
- [11] Diana Ruth-Janneck. 2011. Experienced Barriers in Web Applications and Their Comparison to the WCAG Guidelines. In *Information Quality in e-Health*, Andreas Holzinger and Klaus-Martin Simoncic (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 283–300.
- [12] Antonio Giovanni Schiavone and Fabio Paternò. 2015. An extensible environment for guideline-based accessibility evaluation of dynamic Web applications. *Universal Access in the Information Society* 14, 1 (2015), 111–132. <https://doi.org/10.1007/s10209-014-0399-3>
- [13] Sven Schmutz, Andreas Sonderegger, and Juergen Sauer. 2016. Implementing Recommendations From Web Accessibility Guidelines: Would They Also Provide Benefits to Nondisabled Users. *Human Factors* 58, 4 (2016), 611–629. <https://doi.org/10.1177/0018720816640962>
- [14] Markel Vigo, Justin Brown, and Vivienne Conway. 2013. Benchmarking Web Accessibility Evaluation Tools: Measuring the Harm of Sole Reliance on Automated Tests. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A '13)*. ACM, New York, NY, USA, Article 1, 10 pages. <https://doi.org/10.1145/2461121.2461124>
- [15] W3C. 2018. Web Content Accessibility Guidelines (WCAG) 2.1. Retrieved December 15, 2019 from <https://www.w3.org/TR/WCAG21/>
- [16] W3C. 2019. Accessibility testing. Retrieved December 15, 2019 from https://www.w3.org/wiki/Accessibility_testing
- [17] W3C. 2019. Evaluating Web Accessibility Overview. Retrieved December 15, 2019 from <https://www.w3.org/WAI/test-evaluate/>
- [18] W3C. 2019. <https://www.w3.org/WAI/ER/tools/>. Retrieved December 16, 2019 from WebAccessibilityEvaluationToolsList
- [19] W3C. 2019. Selecting Web Accessibility Evaluation Tools. Retrieved December 15, 2019 from <https://www.w3.org/WAI/test-evaluate/tools/selecting/>
- [20] W3C. 2019. Web Content Accessibility Guidelines (WCAG) Overview. Retrieved December 15, 2019 from <https://www.w3.org/WAI/standards-guidelines/wcag/>